PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/445 — A2

(11) International Publication Number: **WO 00/54149**

(43) International Publication Date: 14 September 2000 (14.09.00)

(21) International Application Number: PCT/US00/06322

(22) International Filing Date: 10 March 2000 (10.03.00)

(30) Priority Data:
60/123,592 10 March 1999 (10.03.99) US

(71) Applicant *(for all designated States except US)*: AUTOMATION CONTROL PRODUCTS LLC [US/US]; Suite 200, 6865 Shiloh Road East, Alpharetta, GA 30055 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: CRANDELL, Matthew, O. [US/US]; 128 Shady Grove Lane, Alpharetta, GA 30004 (US). CAINE, Timothy, A. [US/US]; 270 Mayfield Farms Drive, Lawrenceville, GA 30043 (US). HORTMAN, Matthew, B. [US/US]; 1210 Avalon Drive, Lawrenceville, GA 30044 (US). CANNADY, Randy [US/US]; 385 Twin Brook Way, Lawrenceville, GA 30043 (US). BARKER, Matthew, E. [US/US]; 871 Hampton Hill Court, Lawrenceville, GA 30044 (US). JOHNSON, Thor, M. [US/US]; 1086 Realm Lane, Lawrenceville, GA 30044 (US).

(74) Agents: GRIFFIN, Malvern, U., III et al.; Alston & Bird LLP, P.O. Drawer 34009, Charlotte, NC 28234-4009 (US).

(81) Designated States: AE, AL, AM, AT, AT (Utility model), AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), DM, DZ, EE, EE (Utility model), ES, FI, FI (Utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: METHODS AND SYSTEMS FOR REDUCED CONFIGURATION DEPENDENCY IN THIN CLIENT APPLICATIONS

(57) Abstract

A computer network comprising a client and a server is provided, wherein a client session is initiated. The client executes a local boot image associated with the client and sends a startup request to the server over a computer network. In response to the request, the server transmits a client boot image to the client. The client boot image comprises a client operating system. After receiving the client boot image, the client initiates operation of the client operating system by executing the boot image. The client boot image may also comprise a client-side manager that is installed when the client executes the client boot image. In this embodiment, the client-side manager transmits a configuration request to the server which includes a unique client identifier. The server receives the configuration request and retrieves a configuration profile for the client using the unique client identifier. The server then uses the configuration profile to determine client configuration data and transmits the configuration data to the client. In one embodiment, the client configuration data comprises all of the software necessary for client operation. The client receives the configuration data and the client manager uses the configuration data to configure the client.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHODS AND SYSTEMS FOR REDUCED CONFIGURATION DEPENDENCY IN THIN CLIENT APPLICATIONS

FIELD OF THE INVENTION

The present invention relates to thin client/server systems in a network environment. In particular, the present invention relates to a thin client application with reduced client configuration requirements.

BACKGROUND OF THE INVENTION

During the 1960's, when computers began to become a permanent and widespread feature of the business environment, the accepted model for computing was a large mainframe computer, which acted as a host or server that provided computer services for a large number of dumb terminal clients. This model became the industry standard in the 1970's. The mainframe computers of that era typically contained a large main memory, and extensive repertoire of instructions. The mainframe computers were also very large and expensive. The mainframe's operating system allowed each user to access the mainframe from a terminal having limited hardware and software intelligence on a time-sharing basis. With the high cost/high capability of the host making for a relatively rich server, it only made sense to have low cost/low capability of relatively "thin" clients.

With the advent of the personal computer, and its virtual explosion of popularity following the introduction of the IBM Personal Computer (IBM PC) in 1981, the computing model began to change. Through the 1980's personal computers grew in popularity and the price of computing hardware, including memory, dropped dramatically. The model for computing shifted to accommodate powerful user-centralized systems, as individual users harnessed more and more raw computing power on their desktops. The need for centralized management and control of resources such as printers and certain software still left a place for a host/ or server/client providing access to many clients. However, the clients were no longer dumb terminals, instead the clients were relatively powerful themselves, having large capacity main memory as well as storage memory, usually in the form of a well-

known hard disk. Thus the server/client model commonly became one involving, relatively speaking, "fat" clients.

The 1990's have witnessed a growing popularity of the Internet and the World Wide Web, a multimedia subset of the Internet. The Internet serves as a worldwide network linking computers all over the globe to each other. The 1990's have also witnessed a return to networked environments to facilitate the free exchange of information among users. As the speed at which information can be communicated over a network increases, the ability to maintain centralized control over a networked environment becomes more feasible. It also alleviates the need for networked clients to maintain powerful computing systems. Thus, the client computing model has once again turned to the need to support thin clients. Such modern thin clients have little need for very much electronic or main memory and practically no need for a hard disk.

The computing model for many servers over the Internet or local area network serving a plurality of thin clients is attractive because it reduces the cost of ownership for computer users and reduces the threat of their workstations becoming obsolete due to computer programs requiring more and more memory for execution. The memory demands may now be satisfied at the servers while the same thin clients may be used for years to come.

This is particularly true in the field of industrial automation control where it is important to maintain uniformity among the thin clients and provide a means for easy system-wide upgrades. As a result, many industrial automation systems could be implemented at a lower cost by implementing thin client/server systems. Recent efforts, however, still suffer from the impact of the low-cost personal computer revolution because the thin terminals routinely include unnecessary software and non-volatile memory means such as hard disk drives. The thin clients also typically include an operating system, device drivers, basic software applications, and other components that are stored on the thin clients and increase the costs of the thin client hardware unnecessarily. There is also a very high degree of variation between the hardware and software components in individual thin clients making it difficult for a server to communicate properly with thin clients containing different hardware and software packages. For example, if a server utilizes Windows NT as its operating system, it must be configured to communicate properly with the thin clients in the network. Typically, thin clients contain limited non-volatile memory space which

stores an operating system, hardware device drivers, and basic application software. For example, a thin client may contain the Windows CE operating system and the device drivers necessary to operate the thin clients video card, network interface, monitor, mouse, and keyboard. The Windows NT server and the Windows CE thin client must be configured to communicate with each other. If the thin client is subsequently replaced with another thin client containing different hardware or software, both the server and the thin client must be reconfigured to account for the changes in hardware and software. Thus, once a particular system has been formed, it is difficult to substitute different thin client configurations.

Problems may also arise from changes in the thin client configuration. For example, if the monitor in a particular thin client is replaced with a different size or brand of monitor, the thin client must be reconfigured to account for the change. Thus, every thin client that is fitted with different hardware or software must be reconfigured to account for any changes to the thin client hardware or software. There currently is no method for automatically configuring a thin client, regardless of its hardware configuration, directly from the server.

Thus, in current thin client/server systems, the thin clients limit the flexibility of the networked system because both the server and the client are designed according to the particular hardware and software configuration of the thin clients in the system. The thin client configuration also limits the ability of the server to provide system-wide updates because any updates must remain compatible with the thin client configuration. Thus, there is a need in the industry to provide a thin client/server system that is not constrained by the configuration of the thin client and that affords all of the advantages provided by centralized server control over numerous thin clients. There is also a need for a thin client/server system that is not platform dependent so that different client configurations are easily accommodated.

SUMMARY OF THE INVENTION

The present invention solves many of the problems associated with current thin client/server systems by providing a platform independent means for implementing a computer network that is not dependent on the particular configuration of the thin client. This is accomplished utilizing a thin client that is configured by the server each time it is powered on. Advantageously, the present invention also allows the thin client to be configured remotely from the server thus

centralizing system operation and eliminating many of the problems associated with conventional networked systems such as those resulting from improper changes to the thin client. The hardware configuration of the thin client is also immaterial. As long as the thin client contains enough volatile memory to communicate with the server and execute the programs downloaded from the server, any suitable computer hardware configuration will suffice. The present invention is also advantageous because it decreases the costs associated with systems containing numerous thin clients. The hardware and software requirements for the thin client are minimal because the thin client need only comprise volatile memory and a local boot image.

According to one embodiment of the present invention, a method is provided for initiating a client session in a computer network. The client executes a local boot image associated with the client and sends a startup request to the server over a computer network. In response to the request, the server transmits a client boot image to the client. The boot image comprises a client operating system. After receiving the boot image, the client initiates operation of the client operating system by executing the boot image.

According to an aspect of the invention, the boot image also comprises a client-side manager which is installed when the client executes the client boot image. In this embodiment, the client-side manager transmits a configuration request to the server which includes a unique client identifier. The server receives the configuration request and retrieves a configuration profile for the client using the unique client identifier. The server then uses the configuration profile to determine client configuration data and transmits the configuration data to the client. The client receives the configuration data and the client manager uses the configuration data to configure the client.

According to another aspect of the invention, the configuration data comprises customizable modules that may contain configuration parameters, devices drivers, or software applications for the client.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a high-level representation of the thin client/server network according to one aspect of the present invention.

FIGS. 2A-2B are schematic representations of the memory configuration of a server of FIG. 1 before and after a client session is initiated in accordance with one embodiment of the present invention.

FIGS. 3A-3B are schematic representations of the memory configuration of the thin client of FIG. 1 before and after a client session is initiated in accordance with one embodiment of the present invention.

FIG. 4 is a flow diagram showing the operation of a thin client and the server in accordance with one aspect of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention now will be described more fully hereinafter with reference to the accompanying drawings, in which preferred embodiments of the invention are shown. This invention may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein; rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to those skilled in the art. Like numbers refer to like elements throughout.

The present invention is generally implemented as part of a computer network comprising at least one thin client (hereinafter referred to as either a "thin client" or "client") and at least one server interconnected over a computer network. FIG. 1 represents a high-level overview of an embodiment of the present invention. As shown in FIG. 1, at least one server **10** is connected to at least one thin client **12** over a computer network **14**. Although multiple servers **10** and multiple thin clients **12** are shown in FIG. 1, it will be appreciated that the number of servers and thin clients may vary according to the requirements of the application for which the present invention is being implemented. The number of servers **10**, however, must be sufficient to support the operation of the number of clients **12**. Conventional servers are often capable of supporting up to 50 thin clients so, in its simplest configuration, the present invention will require only a single server. If a particular application requires more thin clients than can be supported by a single server, then a second server should be added. Once the additional server is added (by plugging into the network), the thin clients can be redistributed evenly across the servers. This can be done either manually, where a user configures the server directly to logically attach a certain client to a particular server, or, more probably, by allowing the servers to

automatically share the load as needed. In this way, performance on all of the clients will remain relatively constant. Even if not required, multiple servers may be desired to provide a backup system should one server fail. If the load is shared between servers, the failure of one server can be compensated by shifting the load to the remaining server(s).

Referring still to FIG. 1, all communications between the server **10** and thin clients **12** are routed over the computer network **14**. The server(s) **10** and the thin client(s) **12** are connected to the computer network using standard network cables well known in the art. Both the server **10** and the thin client **12** must include a network connection such as an Ethernet port to facilitate communication over the network. It will be appreciated, however, that numerous alternative network connections may be utilized, such as ATM or Token Ring.

The computer network **14** may comprise any of several well-known network schemes such as the Internet, a local area network, or a wide area network. The configuration of the computer network is unimportant so long as it allows the server(s) to communicate with the thin term(s) as required by the present invention. The network configuration will be determined, in large part, by what type of system is best suited to the particular application of the invention. The computer network may also allow communication from thin client to thin client but such communication is not necessary to practice the present invention.

I. Server Description

Referring still to FIG. 1, the server **10** is a computer with sufficient resources to support the desired number of thin clients. In a preferred embodiment, the server is a computer comprising a 32-bit x86 processor, a hard disk with 128 megabytes (MB) or more of free space, a network interface card, a CD-ROM drive, 32 MB or more of random access memory (RAM) plus an additional 4 to 12 MB or more of RAM for every thin client supported by the server, a mouse, a keyboard, and a monitor. It will be appreciated, however, that this system is provided merely for illustrative purposes. The server may also be a unix-based system such as the Sparcstation provided by Sun Microsystems or any other computer system with sufficient resources to operate as a server. The particular configuration of the server is also unimportant so long as the server contains enough resources to support the thin clients it controls.

Turning now to FIG. 2A, there is shown a schematic diagram representing the functional components of the server prior to the initiation of a thin client session according to one embodiment of the present invention. The server **10** is configured with an operating system **26** and software to facilitate server operation and communication with the thin clients. The server software may be part of the operating system or added as a plug-in or additional software program on top of the operating system. The operating system **26** shown in FIG. 2A is assumed to include the software necessary to facilitate operation as a server. In a preferred embodiment, the server utilizes Microsoft Windows NT Terminal Server Edition or Microsoft 2000 Server operating system. Again, it will be appreciated that this embodiment is merely illustrative and is not intended to be limiting. The operating system need only contain sufficient functionality for the server to operate in a server/client environment and sufficient resources to support the server-side manager (as described below) and the demands of the thin clients.

The server should also contain a network connection protocol (not shown) to facilitate communication over the network **14** (as shown in FIG. 1). Preferably, the server uses TCP/IP protocol. In the embodiment shown in FIG. 2A the network communication protocol agent is assumed to be part of the operating system **26**, but it could also be added as a separate component if necessary. In the preferred embodiment described above, the Windows NT operating system includes the TCP/IP protocol. It will be appreciated, however, that different servers may employ other types of network connection protocols such as NetBeui or NWLink IPX/SPX.

Referring still to FIG. 2A, the server should also contain a client/server communication protocol agent **24** that facilitates communication between the server and the client and allows the client access to the server. The communication protocol agent also facilitates communication of the user's mouse movements and keyboard commands between the thin client and the server. This communication protocol agent may be built into the server and client operating systems or configured as a separate application. Thus, although the communication protocol agent is shown as a separate component in FIG. 2A, it may in operation be included as part of the server operating system **26**.

In a preferred embodiment the server utilizes Citrix Metaframe (or Citrix Device Services), both off-the-shelf products, as the communication protocol agent **24**. The Citrix product uses the Independent Computing Architecture (ICA) protocol

to facilitate remote presentation services and allow the thin client access to the server. The Citrix product also extends the functionality of Microsoft's Terminal Server product with additional client and server functionality, as is well known in the industry. It provides the functionality to make the system responsive enough for true thin client operation.

It will be appreciated, however, that numerous other communication protocol agents may be utilized without altering the novel aspects of the present invention. If the communication protocol agent is built into the client and server operating systems, it may be unnecessary to add an additional protocol if the desired level of client/server functionality is included. For example, if the server is configured with Windows NT and the thin client is configured with Windows CE, these two operating systems contain a built in protocol known as RDP to facilitate server/client communication. In this configuration, a separate communication protocol agent may not be necessary.

The server **10** also contains a server-side manager **22**. The server-side manager **22** comprises computer code that may be located on the server hardware or software that is executed to provide control over the thin clients and oversee initialization and configuration of the thin client **12**. In a preferred embodiment the server-side manager **22** is software resident on the server **10**. The server-side manager **22** is responsible for configuring each thin client in response to a request from the client for configuration data (which may be part of a startup request described further below) to insure that the thin client boots properly. The server-side manager **22** determines the appropriate configuration for the thin client **12** and sends the proper environment variables, device drivers, and device parameters for the thin client. Preferably, the server-side manager **22** also monitors the thin client **12** and allows for various remote functions such as thin client **12** remote reboot and touchscreen calibration.. The server-side manager **22** also allows for auto configuration of initial client terminal settings, drop-in replacement of terminals (with the new terminal receiving the configuration and session of its predecessor) and configuration of terminals using default or group-wide properties, server side load balancing and broadcast boot of clients.

In a preferred embodiment of the invention, the server-side manager **22** includes a graphical user interface (GUI) that allows a user (such as the server administrator) to remotely configure the thin clients from the server **10**. The GUI may also include the details of the networked system such as the number of thin

clients, the working groups in which the thin clients are arranged, and configuration information and other details about the thin clients **12** currently connected to the server **10** such as the client's identifier, network address, and operating system and version.

Referring still to FIG. 2A, the server **10** also comprises a client boot image **20**. This boot image is distinct from the server's own boot image that facilitates start-up of the server **10**. The client boot image **20** comprises the components necessary to initiate operation of the thin client. In a preferred embodiment, the client boot image **20** comprises a client operating system and a client-side manager.

In one embodiment, the client boot image **20** may further comprise I/O drivers which the client **12** may need to communicate with external devices. If the particular client that is booting does not need the I/O drivers included as part of the client boot image, then they are simply not used by that client. The boot image may also contain a client communication protocol agent corresponding to the communication protocol agent **24**, if the communication protocol agent is desired to facilitate communication between the thin client **12** and the server **10**. For example, in the preferred embodiment described above where the server includes the Citrix Metaframe (or Citrix Device Services) communication protocol agent, the client boot image **20** may include a Citrix client capable of communicating with the Citrix software located on the server **10**.

Additionally, the client boot image **20** may contain a client GUI to provide an easy to use interface on the thin client **12**. In a preferred embodiment, the software contained in the client boot image is compressed or packed to facilitate quicker transport of the client boot image over the network to the client **12**. In this embodiment, the client boot image also preferably contains executable commands suitable for decompressing and installing the software. Preferably, there is one client boot image for all of the clients **12** that need to be started; the individual configuration of each client is handled by initialization files (stored on the Server) that are read and processed by the server-side manager.

In a preferred embodiment, the client boot image **20** comprises a Linux kernel. The kernel contains the core portions of the Linux operating system that is installed when the kernel is unpacked executed at the client **12**. Linux is a free computer operating system (derived from UNIX) with an open architecture that allows for easy customization. In this embodiment, the client boot image **20** may also comprise the

X-Windows environment as the client GUI and a Citrix client as the client communication protocol agent. The X-Windows environment is a free GUI developed to provide a windows-based interface for Unix and Linux operating systems. The Citrix client provides client-side support for the Citrix Metaframe communication protocol agent installed on the server as described above. The Citrix client is preferably configured to operate via the X-Window GUI.

After the server has completed the thin client initialization and configuration process (as described below), the server then waits for the client to logon to the server. When the client logs on, the server initiates a client session. FIG. 2B shows the functional components of the server after several client sessions have been initiated. Referring to FIG. 2B, the server allocates a portion of its memory for client workspaces **28** for each client. As more fully discussed below, the client workspaces **28** allow client applications to be run directly from the server.

II. Client Description

The thin client may be any of several commercially available computing products or a specially designed "dumb terminal" suitable for operation according to the present invention. The physical box comprising the client may be smaller than a standard desktop computer, and will preferably include connections for power, a monitor, network, a keyboard and a mouse. Additionally, if a client is being used for input/output (I/O) to a special piece of equipment, that Client will have whatever connections are required to communicate with that equipment. It will be appreciated that the particular physical configuration of the thin client is not important to operation of the present invention so long as the thin client contain the components necessary to provide the functionality described below.

FIG. 3A is a schematic diagram representing the functional components of the thin client **12** before it is powered on. At this stage, the thin client comprises only volatile memory **32,** such as random access memory (RAM), and a local boot image **30** residing in non-volatile memory associated with the thin client, such as read only memory (ROM). The volatile memory **32** is typically random access memory such as the memory provided by SIMM or DIMM circuit chips. The thin client must contain enough volatile memory **32** to allow the thin client's operating system and the software necessary for network communication and interaction to load and run. In a typical configuration, the thin client will contain approximately 32 MB of volatile

memory. It will be understood, however, that this amount of volatile memory is merely illustrative and different applications of the present invention may contain thin clients with significantly more or less volatile memory.

As shown in the embodiment illustrated in FIG. 3A, the local boot image **30** is located on a small physical read-only memory device such as an EPROM that is located within the thin client **12**. The local boot image, however, may be located in numerous locations so long as it is associated with the thin client. In one embodiment, the boot image is included as part of the Basic Input Output System (BIOS) of the client. Alternatively, the local boot image may reside on a network connection component of the thin client **12** such as an Ethernet card. In another embodiment, if the thin client **12** comprises a floppy disk drive, the local boot image may reside on a floppy disk that is inserted into the thin client **12** prior to start-up. It will be appreciated that the local boot image requires minimal memory space and thus may be located in numerous locations. The only requirement is that the thin client **12** be configured such that it locates the local boot image after it is powered-on.

The local boot image **30**, contains executable commands that direct the thin client **12** upon startup. When the thin client **12** receives power, it locates the local boot image **30** and begins to execute the executable commands therein that direct the thin client **12** to contact the server **10** and load the client boot image **20** (as shown in FIG. 2A) located on the server. The client boot image **20** preferably contains instructions for unpacking and installing the software programs (including the client operating system) contained within the client boot image **20**. The local boot image may also contain the necessary commands to begin execution of those instructions. After the client is booted, the local boot image plays no other role until the next time the thin client is powered-on.

Advantageously, the thin client **12** does not need to store any other software, device parameters, or local environment variables. In one embodiment, however, the thin client may store network address information. All of these components are stored on the server **12**. In a preferred embodiment, the thin client contains only the local boot image **30** and volatile memory **32** such as random-access memory (RAM) sufficient to load and execute the software downloaded from the server. When the thin client is powered down, all of the RAM is erased. This configuration is unique because the thin client is automatically configured by the server each time it is powered on. This allows the server to automatically detect changes to the thin client

and properly configure the client without requiring any user action on the client side. It also allows for client replacement, where the replacement client receives a configuration which makes it the functional equivalent of its predecessor.

This configuration is also advantageous because the thin client **12** does not have to be a powerful or complex computer system to operate the programs downloaded from the server **10**. It also does not have to be configured with any particular hardware or software. This affords great hardware flexibility because numerous types of thin client machines may be used ranging from outdated computer systems to specially designed thin terminals.

After the client downloads the client boot image from the server, it stores the software contained in the client boot image into the volatile memory. FIG. 3B shows the functional component of the thin client after the client boot image has been downloaded and the client software has been installed. In the embodiment illustrated in FIG. 3B, the thin client downloads a client operating system **34**, a client-side manager **36**, a communication protocol **38**, and I/O drivers **40**. These components are temporarily stored in the client volatile memory space. As discussed above, in a preferred embodiment, the client operating system **34** is Linux and the communication protocol **38** is a Citrix client. The I/O drivers **40** vary according to the particular configuration of the thin client and the external I/O devices that it communicates with.

The client-side manager **36** communicates with the server-side manager **22** to facilitate configuration and operation of the thin client **10** including initiating download of any software modules and providing an interface for the server-side manager to receive specifics of client performance. In a preferred embodiment, the client-side manager **36** is the client side of the ThinManager™ product developed by Automation Control Products, the assignee of the present invention.

III. Operation Of The Network

For purposes of illustration, the operation of the system will be described from the initial boot-up of the system. Thus, it is assumed that both the server **10** and the thin client **12** are initially powered off. It is also assumed that both the server and the thin client are connected to the computer network as shown in FIG. 1. To simplify the description of the present invention, it will be described with general reference to FIG. 4 which is a flow diagram illustrating the basic operation of the thin client **12**

and the server **10** by providing an illustrative sequence of communications when beginning a client session according to one aspect of the present invention.

The first step in implementing the network is to start-up the server. The server boots up as would any server well known in the art. The server operating system runs just as it would on any server well known in the art and includes server software to allow it to properly function. The server-side manager is also loaded at start-up. If a communication protocol agent is required or desired to facilitate communication between the client and the server (as is the case if the operating systems of the server and thin client do not include a built-in protocol such as RDP), a communication protocol agent, such as Citrix Metaframe, should be loaded on the server at startup as well.

Referring now to FIG. 4, after the server is operational, one or more of the thin clients are powered-on. The following discussion details the startup process for a single thin client but, because the operation of the system is identical for each thin client started, the process followed may be duplicated for as many clients as required. When power is applied to the thin client as shown in block **42** of FIG. 4, the BIOS located on the client's processor searches for a unique code that indicates a local boot image. As described above, depending on the configuration of the thin client, the local boot image may be located in local, non-volatile memory resident on the client or on an Ethernet card, floppy disk, etc. It is assumed that the thin client's BIOS is configured so that it is able to locate the local boot image. In a preferred embodiment, the local boot image is located in a local (E)PROM stored on the thin client. As shown in block **44**, once the local boot image is located, the processor begins executing the code contained in the boot image.

At this time, the server is still unaware of the thin client's existence. Very early in the client's boot process (i.e., its execution of the local boot image commands), the client sends a startup request to the server, shown as block **46**, which may comprise numerous sub-requests. At a minimum, the startup request includes information sufficient to prompt the server to send the thin client's operating system that is stored in the client boot image stored on the server. In a preferred embodiment, the startup request is accomplished by first sending a signal to the server identifying the thin client and requesting a network address for the client. The network address allows the client to connect to the server and communicate with the server over the network. In a preferred embodiment, the network address comprises an Internet

protocol (IP) address. The thin term client can use any one of several protocols to receive an IP address such as Dynamic Host Configuration Protocol (DHCP), Static IP assignment, BootP, and Reverse Address Resolution Protocol (RARP). The remainder of the discussion herein assumes that the DHCP protocol is used but it will be appreciated by those of skill in the art that the other protocols could be easily substituted without altering the present invention.

When the server receives the DHCP request from the client, depicted in block **48**, it determines an IP address for the thin client. In this embodiment, the server assigns the thin client an available IP address, but it will be understood that other methods may be employed if a different protocol is used. Once an IP address is determined, the server transmits the IP address with the IP address of the server where the client boot image is stored and transmits this information back to the thin client. Once the client receives its network address (IP address) and the location of the client boot image at the correct server, the client transmits a signal to the server requesting that the server send the client boot image, which preferably contains all of the software that the client needs to operate, including the client operating system. It will be understood, however, that the request for the bootable image may be included as part of the initial request for a network address. Regardless, the request for the client boot image is considered part of the startup request for purposes of the illustration in FIG. 4. The server responds to the client's request by transmitting the client boot image to the client as shown by block **50** in FIG 4. The transfer of the client boot image may be accomplished using any of several well-known file transfer methods, but is preferably achieved using Trivial File Transfer Protocol (TFTP). It should be noted however, that the client boot image does not necessarily reside locally at the server but may reside at a location remote from the server from which it is sent to the client. Accordingly, any other references herein to data being sent by the server or located on the server may optionally be located at or sent from a location remote to the server as is well known by those skilled in the art.

In a preferred embodiment, the client boot image is compressed. Thus, when the client finishes downloading the client boot image it must decompress (unpack) the boot image and create the necessary memory space in available volatile memory to store the software contained in the client boot image in local volatile memory. The commands required to decompress the client boot image may be contained within the client boot image or included as part of the thin client's local boot image. After the

client boot image has been downloaded from the server and stored in local volatile memory, the thin client begins executing the executable commands contained in the client boot image.

In one embodiment, the client boot image comprises a Linux kernel which is executed to initiate operation of the thin client's operating system. The Linux kernel sets up a local RAM disk (i.e., a logical file system) on the thin client and unpacks (decompresses) the rest of the client boot image which comprises all of the programs necessary for thin client operation. In this embodiment, the Linux kernel installs a file system into the RAM and executes the startup scripts found in the client boot image to configure the local environment variables such as the thin client's network connection device, local sound or video cards, and any other local devices. In this embodiment, the client boot image also preferably contains a client-side manager which is installed when the client boot image is executed. The client boot image may also contain I/O drivers. If the thin client is connected to an I/O device, the Linux system is also configured to install the I/O drivers need to facilitate communication between the I/O device and the thin client.

If the system requires the use of a communication protocol agent, the client boot image may also contain the software necessary to install the protocol on the client. The client boot image may also contain the client GUI. In one embodiment, the client GUI is an X-windows environment and the communication protocol agent is a Citrix client which is installed as part of the client boot image and which communicates with the Citrix software on the server to transfer all of the video displays, mouse movements, keyboard activity and any data required to make the thin client appear to the user to be a Windows NT session.

In an alternative embodiment, the client boot image contains only a Linux kernel and a client-side manager. In this embodiment, the Linux kernel contains only the core Linux operating system and does not include local environment variables or local device drivers and configuration information. In this embodiment, as illustrated by block **54**, the core Linux kernel is executed to initiate the basic operating system. Additionally, the client-side manager is also installed when the client boot image is executed. The client-side manager is then used to communicate with the server to complete configuration of the thin client.

To accomplish this, the client-side manager transmits a configuration request to the server as depicted in block **56**. The configuration request contains a unique

client identifier which identifies the client to the server. The unique client identifier may be associated with the client as part of the local boot image which is then passed on to the client-side manager after it is loaded. Alternatively, the unique identifier may be determined by the client-side manager. In a preferred embodiment, the unique client identifier is the thin client's MAC (Media Access Control) address (i.e., hardware address). The MAC address is unique to each device attached to a network, is stored in non-volatile memory associated with the client, and is used to identify each network device for the purpose of directing network traffic.

The server receives the configuration request at block **58** and uses the unique client identifier to retrieve a configuration profile for the thin client as shown in block **60**. Preferably, the configuration profile contains information concerning all of the drivers, device parameters, local environment variables, etc. that are necessary to configure the thin client. The configuration profile may also contain information concerning the client communication protocol agent, GUI, or any other software necessary for client operation. The configuration profiles for the thin client may be stored on the server in a look-up table. Using the unique client identifier, the server can locate the configuration profile for a particular client and obtain a list of the configuration data (i.e., software, drivers, device parameters, local environment variables, etc.) necessary to complete initialization and configuration of the thin client. The configuration data is then transmitted by the server to the client as shown in block **62**. The client receives the configuration data and the client-side manager configures the thin client using the data as shown in block **64**.

In one embodiment, the configuration data may also include any additional software necessary for particular thin client applications. This includes any necessary I/O drivers needed by the thin client to communicate with any external devices. These drivers will initialize the external I/O device (regardless of whether it is connected through a serial port, a card slot, or any other mechanism) initiate communication between the client and the I/O device. At this point the I/O device is ready to communicate with the I/O server located on the server (as described below). The configuration data may also include any other software necessary for client operation. This may include a client communication protocol agent, a client GUI, and any software applications used by the client for particular applications.

In another embodiment of the present invention, the thin client **12** is configured using software modules. In this embodiment, the configuration profile for

the client includes a list of the modules that the client requires for proper configuration. Each module is specifically designed for a particular thin client configuration. For example, if the thin client contains a 17 inch monitor with a touchscreen, modules for that client might include one module containing all of the device drivers and parameters for the monitor and another module containing the device driver and parameters for the touchscreen. Additionally, hardware devices added at a later date to the client 12 can be accommodated by simply downloading modules for that device. Modules may also comprise the client GUI, the communication protocol agent, or particular software applications for the client.

This is advantageous because multiple device drivers and configuration parameters need not be included in the client boot image. The client boot image need only contain the basic core operating system for the client and the client-side manager. The complete thin client configuration is then performed by downloading the appropriate modules using the client-side and server-side managers. The scope of potential modules is enormous and will vary from application to application. For example, any I/O drivers or parameters needed by the client may be included in a module. Additionally, the client communication protocol agent, client GUI, or any other software applications used by the client may be included in one or more modules. Modules also provide a high degree of flexibility. For example, if a particular system uses two basic configurations for its thin clients, a module can be developed which contains all of the local environment variables, device drivers, device parameters, etc. necessary for each type of client. Thus, the client is configured automatically upon startup and the only the software necessary for client operation is downloaded onto the client.

In an embodiment of the present invention, the server-side manager communicates with the client-side manager to facilitate easy ("drop in") replacement of terminals. In this embodiment, the server-side manager detects the addition of a new client when more than one of the previously configured clients has dropped offline. In this embodiment, the client-side manager transmits a configuration request to the server-side manager. If the server-side manager detects an unrecognized unique client identifier, it responds with a list of all of the offline clients. The client-side manager then allows the user (local to the client) to indicate the client that is being replaced. The server-side manager then sends the configuration data and programs that were being used by the preceding client and associates the

configuration of the preceding client with the new client base on the unique client identifier of the new client.

In another embodiment, the server-side manager communicates with the client-side manager to facilitate default configuration of multiple thin clients. In this embodiment, the server-side manager sends a configuration to each unknown client (when it boots) based on properties as described in a default configuration profile. The default configuration profile may be stored on the server or retrieved from another location on the network. The default configuration may also be stored in a look-up table under a default client identifier. For unknown terminals that are booted and designated as members of a group of already configured clients, the server-side manager creates a new configuration by inheriting configuration parameters from properties common to the clients in that group. This may be achieved by comparing the configuration profiles of the clients connected to the group or by using a configuration profile that is uniquely associated with the group itself.

At this point, the thin client is booted and it now waits for a stimulus from a user. FIG. 3B shows the functional components of the thin client after it has been booted according to one possible configuration. It will be appreciated that all of the software residing on the thin client including the operating system **34**, client-side manager **36**, communication protocol agent **38** and I/O drivers **40** are resident in volatile memory and are not permanently stored on the thin client. When the thin client is powered down, all of these components will be erased from the volatile memory of the thin client; only the local boot image **30** remains present in the non-volatile memory of the thin client after it is powered down (if the local boot image is initially stored there). Also, it will also be understood that the components shown in FIG. 3B are merely illustrative and are not required for every application of the present invention. The actual components resident in the volatile memory of the thin client will depend on the particular application for which it is used.

Operation from this point on is similar to the operation of conventional client/server systems. Typically, a user attempts to log-on to the thin client and the client transmits a login request to the server. In a preferred embodiment of the present invention, this is achieved using the Citrix communication protocol agent. Once a user attempts to log-on to the client, the Citrix software sends a message to the server, and the server processes this request and provides the Citrix services to the client. The client starts a Windows Terminal Session and the user is now able to begin

normal activity on the client. As discussed above, any of several well-known communication protocol agents may be used to facilitate user login at the thin client.

FIG. 2B shows the functional components of the server after several thin clients have started. As in shown in FIG. 2B, the server retains all of the functionality it contained prior to initiating the thin client session. However, after the client initialization process is complete, the server allocates workspace **28** for each thin client connected to the server. The server uses this space to load any software needed by the client as determined by the particular client application.

If the thin client is being used for local I/O, then the server will also start an I/O server either in the client workspace **28** or in a common area of the system memory. The I/O servers will transfer data and requests from the server to the local I/O devices located on the thin client. The I/O server is also used to make the data read locally by the client available to the server and any other client that needs access to that I/O device. Advantageously, the server controls the local I/O operations on the client through the I/O servers located on the server. This allows the data collected from the local I/O devices to be stored on the server. Thus, if a particular client is shut down, the data collected is not lost because it is maintained on the server.

The client workspace **28** is also used to provide any additional services required by the client such as HMI (Human Machine Interface) software, Word Processors, etc. These programs are started in the client's session on the server and run directly from the server. Advantageously, these programs continue to run in the client's workspace on the server regardless of whether or not the thin client is powered down. Thus, if a task running in the thin client session is being used to control or monitor equipment, power interruption on the client will not disrupt this operation because the server will continue running the operation.

IV. Particular Application Of The Present Invention

The present invention is especially suited for application in factory automation systems. The present invention allows manufacturers to use a very low cost workstation that requires no operator configuration, can be powered down and replaced at any time (without interrupting the manufacturing process), and allows for a single copy of all of the factory control software.

Individual clients are placed at equipment as needed and connected to a server, which is preferably located away from the manufacturing environment and in a

secure, clean computer room. The server is also managed by the company's IT department, allowing easy upgrades and reliable backups. As all of the operator interface screens for the individual clients are located in a single copy on a single machine (the server), the client software can be easily updated and is automatically distributed.

Additional clients are added by simply running a network cable from the nearest hub, plugging in the new client and turning it on. The server detects the new client and sends it the client boot image required to have it up and running in moments without any prior configuration of the new client. If the client load becomes to heavy for the installed server, an additional server can be added which will then be able to share the load with the original server. Additional clients and servers can continue to be added in this manner as required.

For high reliability installations (frequently found in the manufacturing industry), a user can start with more servers than would be required for a standard installation, and use the load sharing capabilities of the server-side software to shift all of the client load to the backup server in the event of primary server failure. The failed server can then be replaced at a convenient time. With the low cost and high reliability of the thin clients, they can also be used in the manufacturing environment to simply perform distributed data acquisition, achieved by using the I/O local to the client. This saves the manufacturer from having to run special communications cables – the data can be gathered by the thin client located near the equipment and then transmitted over a standard network back to the central server. Likewise, data and commands can be sent from the server for equipment local to any client.

The present invention is also well-suited for implementation in office and school environments due to its relatively low cost, easily distributed applications, centrally managed software and easy replacement and addition of clients.

When implemented over any type if Wide Area Network (including the Internet), the present invention can be used as an application service provider (ASP) to distribute software to clients on an as needed basis. The customer, running a thin client, connects to a managed server at a remote location and requests the use of any in stock software product. The software is instantly turned on for that client, and the client can use it just as if he had installed the product locally. Once the software is no longer needed, the client indicates that he has finished with it and the software is deactivated for his client. The time that the software product was used is recorded,

and the user is billed a predetermined rate for use of the software. The customer can get any software package available without having to purchase and install the software on his local machine, but is able to run this package as if he had actually installed it on his local machine.

This mechanism allows for a supervisory program running at the remote location (considered the centrally managed site) to control and monitor the use of any software that it has available. In one embodiment, several servers would be used to serve customers located in various locations. The server at each customer's site accesses, through the Internet or a direct connection, the server at the central site whenever it is in need of new or updated software. The supervisory management program running on an ASP server then verifies and records the customer's identification, along with the current time and date, and then makes the software available to the customer either directly (where the customer runs the software directly on the ASP server) or by setting it up on a local server at the customer's site. It also records the number of users that the customer needs, and records this information in the customer's file.

Many modifications and other embodiments of the invention will come to mind to one skilled in the art to which this invention pertains having the benefit of the teachings presented in the foregoing descriptions and the associated drawings. Therefore, it is to be understood that the invention is not limited to the specific embodiments disclosed and that modifications and other embodiments are intended to be included within the scope of the appended claims. Although specific terms are employed herein, they are used in a generic and descriptive sense only and not for purpose of limitation.

THAT WHICH IS CLAIMED:

1. In a computer network comprising a thin client connected to a server, a method for initiating a client session comprising:

executing a local boot image associated with the client;

sending a startup request from the client to the server;

receiving at the client a client operating system in response to said startup request;

initiating operation of the client operating system.

2. The method of claim 1, further comprising:

receiving at the client a client-side manager in response to the startup request;

using the client-side manager to transmit a configuration request from the client to the server, wherein the configuration request includes a unique client identifier;

receiving client configuration data at the client in response to the configuration request;

configuring the client using the configuration data.

3. The method of claim 2, wherein receiving client configuration data in response to the configuration request includes receiving client configuration data comprising one of a local environment variable, a device driver, and a device parameter.

4. The method of claim 2, wherein receiving client configuration data in response to the configuration request includes receiving client configuration data comprising at least one client module.

5. The method of claim 1, further comprising receiving at the client a communication protocol agent in response to the startup request, wherein the communication protocol agent facilitates communication between the client operating system and the server operating system.

6. In a computer network comprising a client connected to a server, a method for initiating a client session comprising:

receiving at the server a startup request from the client;

transmitting a client operating system to the client in response to the startup request.

7. The method of claim 6, further comprising:

receiving at the server a configuration request from the client, wherein the configuration request includes a unique client identifier;

retrieving a configuration profile for the client using the unique client identifier;

using the configuration profile to determine client configuration data; and

transmitting client configuration data to the client.

8. The method of claim 7, wherein using the configuration profile to determine client configuration data includes using the configuration profile to determine client configuration data comprising one of a local environment variable, a device driver, and a device parameter.

9. The method of claim 7, wherein using the configuration profile to determine client configuration data includes using the configuration profile to determine client configuration data comprising at least one client module.

10. The method of claim 6, further comprises transmitting a communication protocol agent to the client in response to the startup request, wherein the communication protocol agent facilitates communication between the client operating system and the server operating system.

11. A computer network, comprising:

a server comprising a client boot image that includes a client operating system;

a client comprising a local boot image, wherein the client executes the local boot image when powered on, and wherein the local boot image initiates a request that the client boot image be downloaded from the server to the client and initiates operation of the client operating system on the client; and

a network that communicatively connects the client to the server.

12. The computer network of claim 11, wherein the client further comprises means for transmitting a configuration request to the server and for configuring the client using configuration data received in response to the configuration request.

13. The computer network of claim 11, wherein the server further comprises means for determining client configuration data in response to a configuration request from the client.

14. In a computer network comprising a client connected to a server, a method for initiating a client session comprising:

sending a startup request from the client to the server;

receiving at the client a client-side manager in response to the startup request;

using the client-side manager to transmit a configuration request from the client to the server, wherein the configuration request includes a unique client identifier;

receiving client configuration data at the client in response to the configuration request;

configuring the client using the configuration data.

15. The method of claim 14, wherein receiving client configuration data in response to the configuration request includes receiving client configuration data comprising one of a local environment variable, a device driver, and a device parameter.

16. The method of claim 14, wherein receiving client configuration data in response to the configuration request includes receiving client configuration data comprising at least one client module.

17. In a computer network comprising a client connected to a server, a method for initiating a client session comprising:

receiving at the server a configuration request from the client, wherein the configuration request includes a unique client identifier;

retrieving a configuration profile for the client using the unique client identifier;

using the configuration profile to determine client configuration data; and transmitting client configuration data to the client.

18. The method of claim 17, wherein using the configuration profile to determine client configuration data includes using the configuration profile to determine client configuration data comprising one of a local environment variable, a device driver, and a device parameter.

19. The method of claim 17, wherein using the configuration profile to determine client configuration data includes using the configuration profile to determine client configuration data comprising at least one client module.

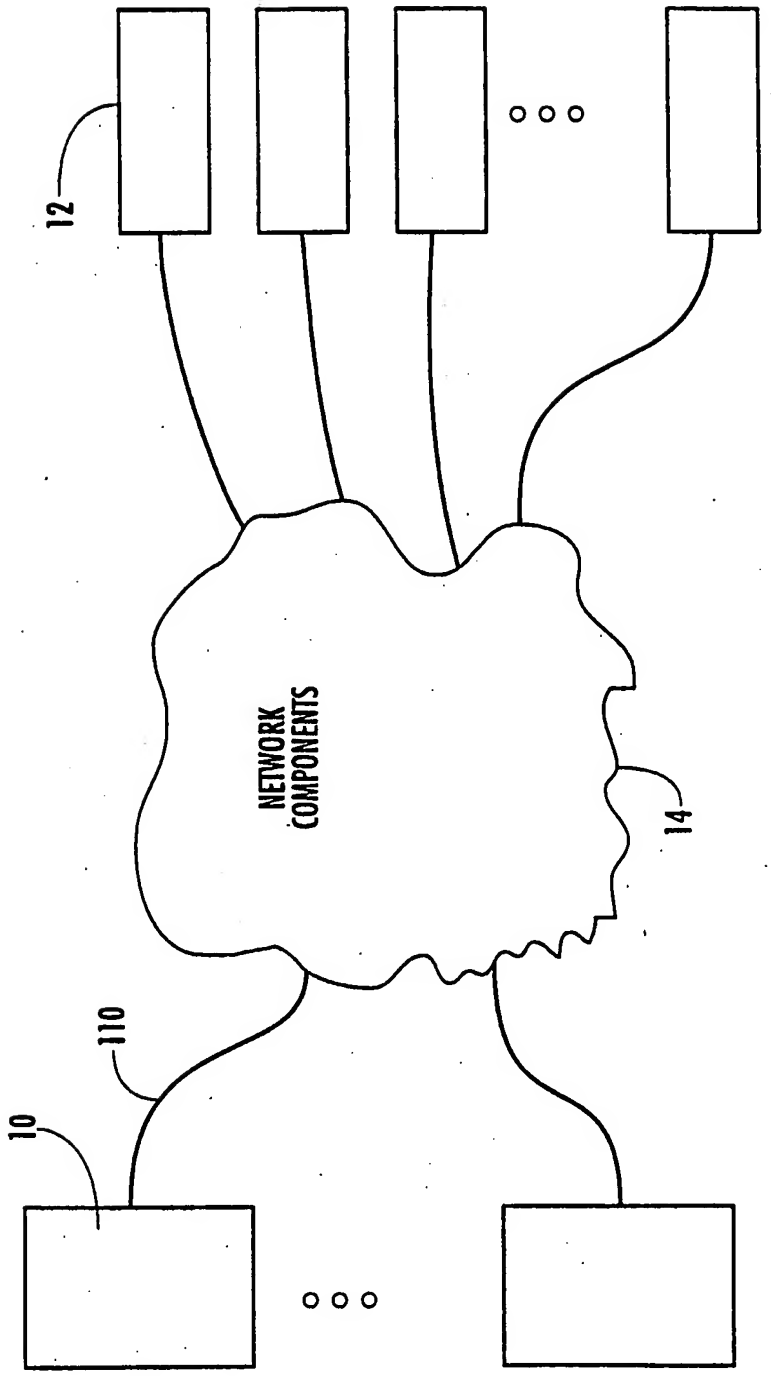

FIG. 1.

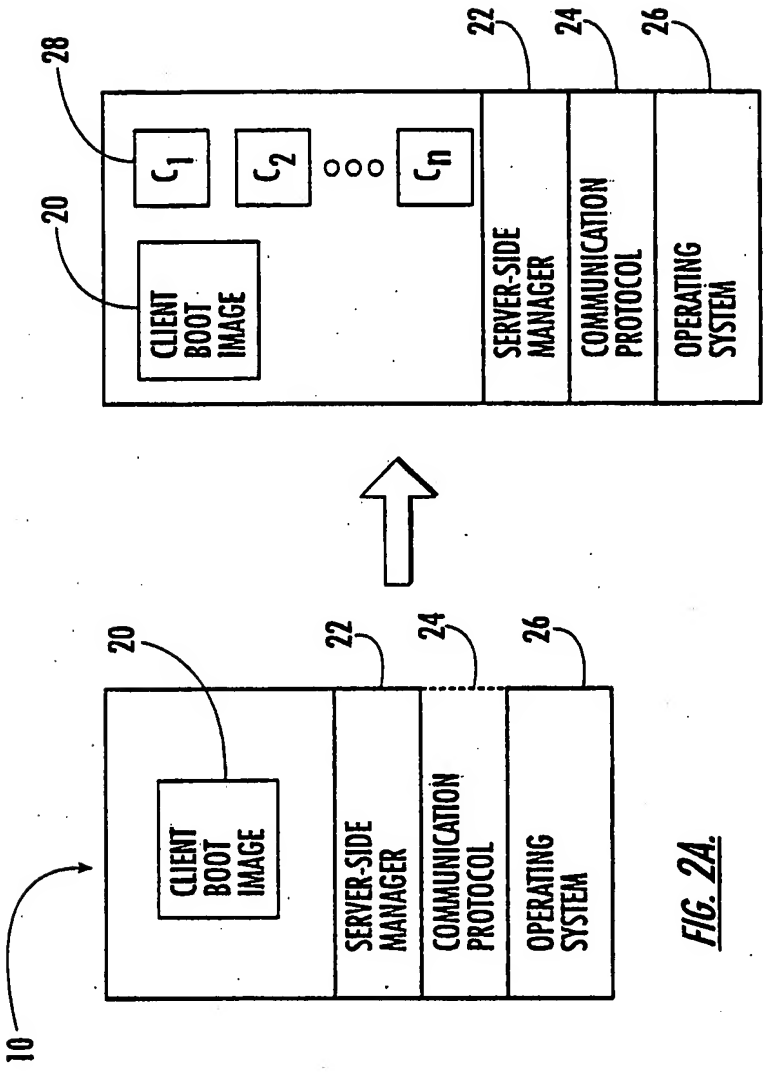

FIG. 2B.

FIG. 2A.

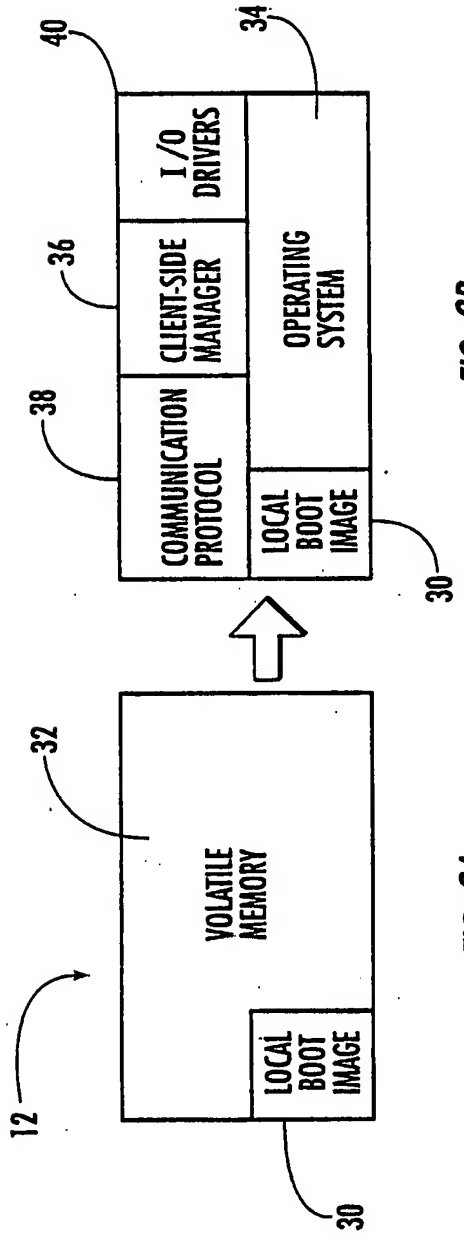

FIG. 3B.

FIG. 3A.

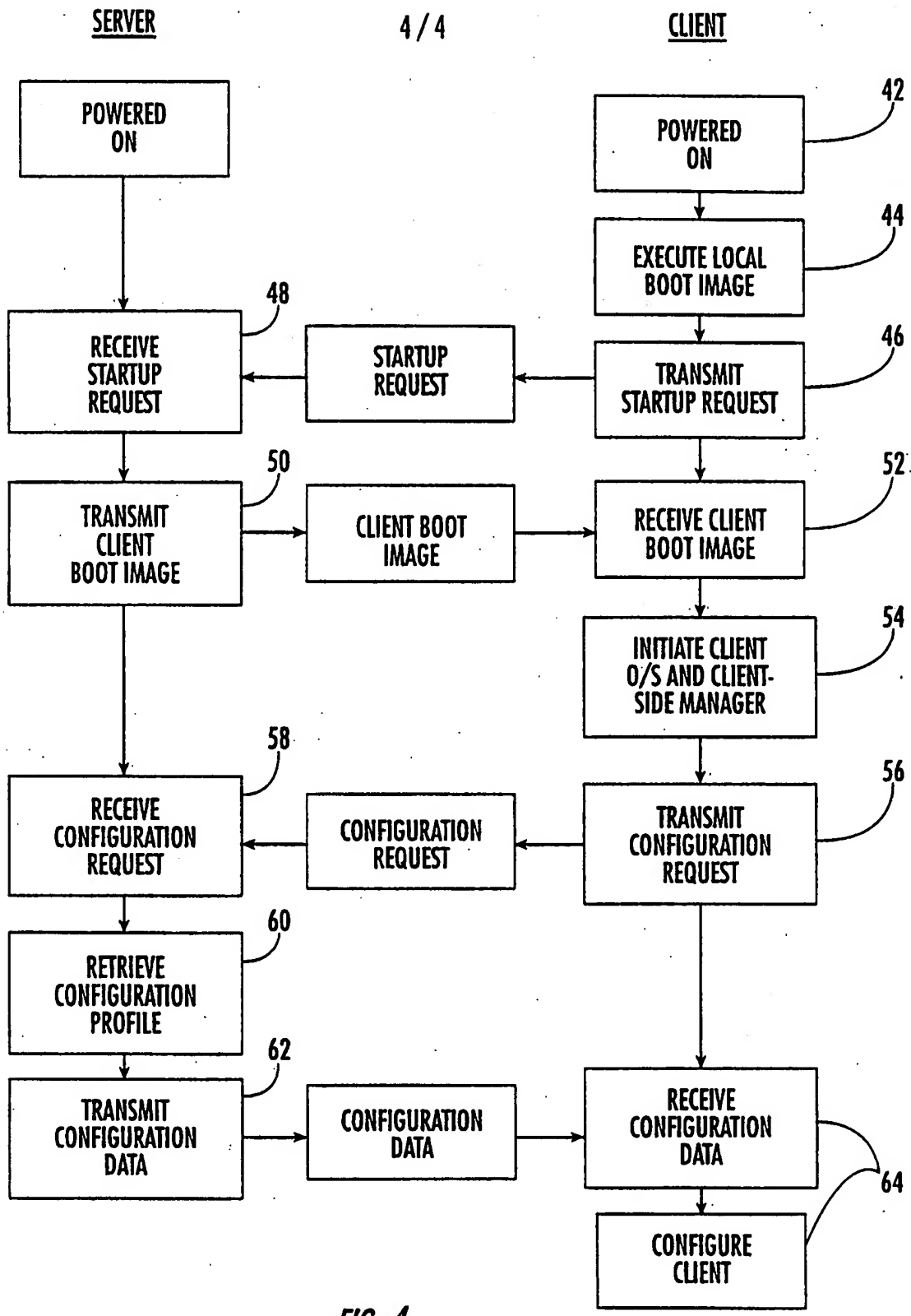

FIG. 4.